Predicting future agent motions for dynamic environments

Predicting Future Agent Motions for Dynamic Environments

Fabio Previtali
Sapienza University of Rome

Alejandro Bordallo
University of Edinburgh

Luca Iocchi
Sapienza University of Rome

Subramanian Ramamoorthy
University of Edinburgh

Abstract*— Understanding activities of people in a monitored environment is a topic of active research, motivated by applications requiring context-awareness. Inferring future agent motion is useful not only for improving tracking accuracy, but also for planning in an interactive motion task. Despite rapid advances in the area of activity forecasting, many state-of-the-art methods are still cumbersome for use in realistic robots. This is due to the requirement of having good semantic scene and map labelling, as well as assumptions made regarding possible goals and types of motion. Many emerging applications require robots with modest sensory and computational ability to robustly perform such activity forecasting in high density and dynamic environments. We address this by combining a novel multi-camera tracking method, efficient multi-resolution representations of state and a standard Inverse Reinforcement Learning (IRL) technique, to demonstrate performance that is better than the state-of-the-art in the literature. In this framework, the IRL method uses agent trajectories from a distributed tracker and estimates a reward function within a Markov Decision Process (MDP) model. This reward function can then be used to estimate the agent's motion in future novel task instances. We present empirical experiments using data gathered in our own lab and external corpora (VIRAT), based on which we find that our algorithm is not only efficiently implementable on a resource constrained platform but is also competitive in terms of accuracy with state-of-the-art alternatives (e.g., up to 20% better than the results reported in [1]).

I. Introduction

Tracking multiple agents in a dense environment, be it humans or robots, is a challenging problem. A multitude of solutions exist for dealing with its different facets, such as overcoming occlusion or motion prediction. Inferring future motion of agents is useful not only for improving tracking accuracy, but also for planning in interactive tasks. Inferring future actions of an agent is known as *activity forecasting*.

Over the past decade, many novel methods have been developed for activity forecasting. These methods include those based on classifiers with structured outputs, which directly discriminate at the level of trajectories albeit with richer representations of the same. Another approach, called Inverse Reinforcement Learning, posits that the motion may be well described as being generated by optimisation within an MDP model, so that learning the motion is the same as inferring the implicit reward function being optimised. In order to keep these algorithms efficient, one often makes assumptions, such as that the scene and environment have been labelled in a semantically meaningful way (thus reducing the sample complexity) or that the space of possible motions is well parametrised and understood.

In many realistic robotics domains, we must apply these methods with a less clear understanding of potential goals (perhaps because the environment is new to the robot, such as in a rescue or rapidly changing construction environment),

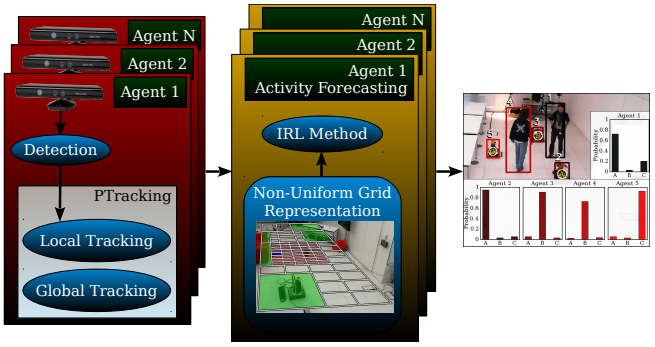


Fig. 1. Activity forecasting using the proposed framework based on IRL. Raw data are acquired from a set of Kinects and then processed by the PTracking algorithm. The agent trajectories in output are subsequently used by an IRL method to generate the agent's set of possible future activities.

including in dynamic environments [2]. We address this setting by combining a distributed multi-camera tracking method and a multi-resolution representation of the environment, with a standard IRL method. This paper uses a relatively standard IRL method, allowing for the possibility that more sophisticated alternatives could be employed in future, within the overall proposed framework. Our focus is on demonstrating that a complete pipeline including this IRL method can be implemented in a resource constrained real-time setting, to solve the challenging problem of predicting agent's future motion.

Contributions. We propose an integrated framework (Fig. 1) that brings together an IRL technique with distributed tracking and multi-resolution state representation, such that it (1) does not rely on semantic scene labelling before activity forecasting, (2) incrementally updates the IRL model over time as new data becomes available and (3) makes use of non-uniform grids for state representation, making the entire framework linearly scalable with respect to the size of the environment. We use *PTracking*, our distributed multi-camera multiple object tracker, as the first component [3]–[5]. Local and global agent position, as well as velocity estimates, are updated via online tracking, providing trajectories that are then used within an IRL algorithm. This algorithm is fairly standard, and taken directly from the literature, in this first instantiation of our framework. While this allows us to implicitly consider sensor noise and false positives, we do recognise that a more elaborate IRL method that can target a Partially Observable Markov Decision Process motion model - something that is still hard to do and certainly not yet efficient for robot implementation - could be a useful future step. The output of this process is a set of reward functions - one for each goal (see Section IV-B) - per agent,

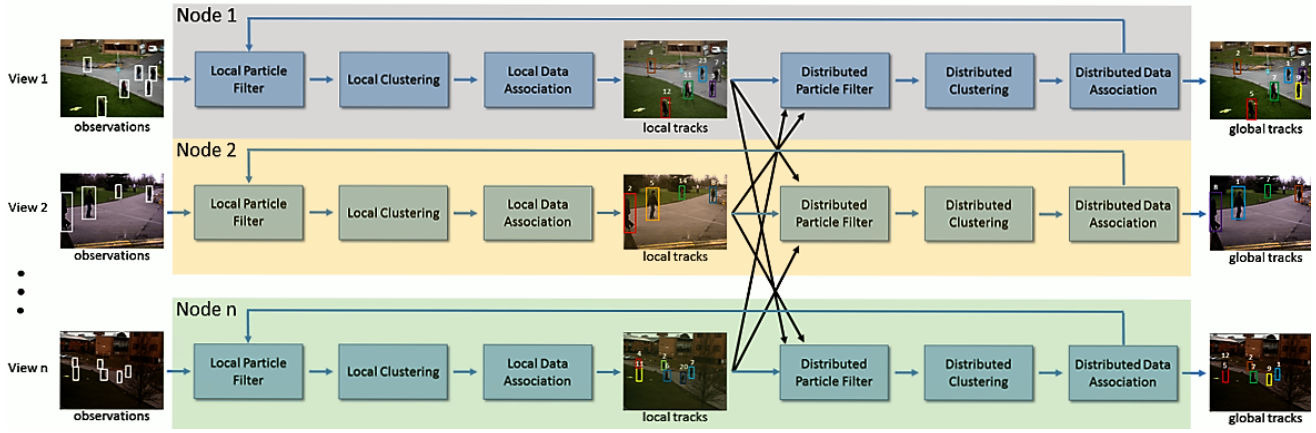


Fig. 2. The functional architecture of the *PTracking* approach. Each node performs a local and a global estimation of the observed situation. For the global estimation, local information collected from other nodes (black arrows) are used.

describing its goal-oriented navigation policy across space. These reward functions represent the agent's set of possible future activities, forecasted or chosen from a probability distribution function via comparison with real-time observed agent behaviour. Additionally, the proposed approach allows us to perform anomaly detection by adopting a flexible notion of trajectory distance, such as the *Fréchet distance* which allows for graceful degradation even when dealing only with small fragments of the overall motion.

We show that our method infers agent goals accurately in a varied set of environments. We present a comparison of our framework against state-of-the-art alternatives from the literature. We use as baselines, a method of Ziebart *et al.* [1], the *Maximum Entropy Markov Model* (*MEMM*) and a random walk method. A quantitative evaluation using multiple data sets shows how our approach performs competitively.

II. Related Work

The problem of understanding activities of people in a monitored environment is becoming increasingly well studied by researchers in multiple communities. The focus is often on two types of challenges - *human activity classification* and *activity recognition*. Human activity classification is an important yet difficult problem in computer vision [6], whose aim is to determine what people are doing, given a set of observations. It finds wide applicability in video surveillance [7], human-computer interfaces [8], sport video analysis [9] and content-based video retrieval [10].

Activity recognition, on the other hand, has as its goal the estimation of a belief-state from observations over time. It follows then that, activity recognition is a temporal classification problem; an agent must generate a sequence of labels, identifying the roles or behaviours of the other agents, given a sequence of observations [11]. Typically methods addressing both activity classification [12] and activity recognition [11] require a substantial amount of captured data during the training phase to generate useful models, a requirement that is not satisfied in many realistic applications. Additionally, even when such data is available, such frameworks may not be able to adapt the learnt model to changes in the monitored environment or, as is needed in some cases, adapt to the dynamics of the environment.

The focus of our work is on trajectory-based human activity analysis. As a proof-of-concept, we propose a *Trajectory-Based Inverse Reinforcement Learning* method for estimating future actions of people (or of robots, vehicles and other agents) from noisy visual input. Similar work has been carried out with success by Ziebart *et al.* [1], who report on activity forecasting by combining an IRL algorithm with a semantic representation of the scene. It is not clear how that technique would deal with unstructured, and in particular not previously labelled, environments.

III. Tracking and Data Association

The problem of tracking multiple objects by using multiple sensors can be formalised as follows. Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ be the set of all moving objects, each one having a different identity, and $\mathcal{S} = \{s_1, \ldots, s_S\}$ be the set of arbitrarily fixed sensors, each one having limited knowledge about the environment (i.e., each camera can monitor only part of the scene). Moving objects are detected by a background subtraction algorithm and the number of objects n is unknown and can change over time. The set of measurements about the objects in the field-of-view of a camera $s \in \mathcal{S}$ at a time t is denoted by $z_{s,t} = \{z_{s,t}^{(1)}, \ldots, z_{s,t}^{(l)}\}$, $l \in \mathbb{Z}$, where a measurement $z_{s,t}^{(i)}$ represents the Cartesian position, width and height of the moving object and it can be either a real object present in the environment or a false positive. The set of all the measurements gathered by all cameras at time t is denoted by $z_{\mathcal{S},t} = \{z_{s,t} \,|\, s \in \mathcal{S}\}$. The history in time of all the measurements coming from all cameras is defined as $z_{\mathcal{S},1:t} = \{z_{\mathcal{S},j} : 1 \leq j \leq t\}$. It is worth noticing that, we do not assume the measurements generated by the cameras to be synchronised.

The goal is to determine an estimation of the positions $\boldsymbol{x}_{s,t} = \{x_{s,t}^{(1)}, \ldots, x_{s,t}^{(v)}\}$, $v \in \mathbb{Z}$, for all the objects in the scene at time t in a distributed fashion - i.e., exploiting all the available sensors. In order to achieve this goal, a possible solution is to use the Bayesian Recursive Estimation, defined as follows:

$$p(x_{s,t}|z_{\mathcal{S},1:t}) = \frac{p(z_{\mathcal{S},t}|x_{s,t})p(x_{s,t}|z_{\mathcal{S},1:t-1})}{\int p(z_{\mathcal{S},t}|x_{s,t})p(x_{s,t}|z_{\mathcal{S},1:t-1})dx_{s,t}} \quad (1)$$

$$p(x_{s,t}|z_{\mathcal{S},1:t-1}) = \int p(x_{s,t}|x_{s,t-1})p(x_{s,t-1}|z_{\mathcal{S},1:t-1})dx_{s,t-1} \quad (2)$$

Eq. (1) and (2) represent a global recursive update that can be computed if and only if complete knowledge about the environment is available - i.e., $p(z_{\mathcal{S},t} \mid x_{s,t})$. Since this is not the case, we approximate the above exact optimal Bayesian computation by means of a Distributed Particle Filter-based algorithm (Algorithm 1). In particular, we extend to a multi-sensor scenario the PTracking method, which is an open-source tracking algorithm based on a Distributed Multi-Clustered Particle Filtering.

The estimation of the positions $\boldsymbol{x}_{s,t}$ is given by the vectors $(\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t})$ containing information about the identity ($\boldsymbol{I}$), the weight ($\boldsymbol{\Lambda}$), the mean ($\boldsymbol{M}$) and the standard deviation ($\boldsymbol{\Sigma}$) of each object, represented as a Gaussian Mixture Model (GMM). The size of the vectors can vary during the execution of the tracking algorithm, depending on the number of detected objects.

The estimation process is made of three main steps: (1) the prediction step, which computes the evolution of the estimations $\boldsymbol{x}_{s,t}$ given the observations $\boldsymbol{z}_{s,t}$ provided by the sensors, (2) the clustering step, which groups the estimations determining their GMMs parameters and (3) the data association step, which assigns each observation to an existing track by considering the history of all existing tracks.

Prediction. The particle filter uses an initial guessed distribution, based on a *transition state* model. Then, using the previous state $\boldsymbol{x}_{s,t-1}$, the transition model, given by the measurements $\boldsymbol{z}_{s,t}$, is applied. From this guessed distribution, a set of samples is drawn and weighted exploiting the current observation $\boldsymbol{z}_{s,t}$. Finally, the *Sampling Importance Resampling* (*SIR*) principle is used to re-sample the particles, which are then clustered to determine the parameters of the final GMM model.

Clustering. A novel clustering algorithm, called *KClusterize*, is used for the clustering phase. KClusterize is designed for fulfilling the following requirements: (1) the number of objects to detect is not known *a priori*, (2) low computational load is needed for real-time applications and (3) each cluster has to reflect a Gaussian distribution. First, the particles are grouped into clusters. Then, a validation step is applied to verify that each cluster actually represents a Gaussian distribution. All the non-Gaussian clusters are split (if possible) in Gaussian clusters. In order to check that a cluster represents a Gaussian distribution, we first draw - for each cluster - a Gaussian distribution centered on the cluster centroid with $\sigma = 1$. Afterwards, we compare the set of points contained into each cluster with the corresponding ones generated from the cluster centroid by using the Euclidean distance. It is important to note that, the final number of Gaussian distribution components provided as output can be different from the one found during the first step. Finally, the obtained clusters form a GMM set $(\boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$ representing the estimations performed by the sensor s at time t.

Algorithm 1: PTracking

Input: perceptions $z_{s,t}$, local track numbers $i_{s,t-1}$, global track numbers $I_{s,t-1}$

Data: set of local particles $\tilde{\xi}_{s,t}$, set of global particles $\tilde{\xi}_{\mathcal{S}',t}$, local GMM set $\mathcal{L}$, global GMM set $\mathcal{G}$

Output: global estimations $x_{s,t} = (\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t})$

begin
- $\tilde{\xi}_{s,t} \sim \pi_t(x_{s,t}|x_{s,t-1}, z_{s,t})$
- Re-sample by using the SIR principle
- $\mathcal{L} = KClusterize(\tilde{\xi}_{s,t})$
- $(\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t}) = DataAssociation(\mathcal{L}, i_{s,t-1})$
- Communicate belief $(\boldsymbol{i}_{s,t}, \boldsymbol{\lambda}_{s,t}, \boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$ to other agents

end

begin
- Collect $\mathcal{L}_{S'}$ from a subset $\mathcal{S}' \subseteq \mathcal{S}$ of cameras within a Δt
- $\tilde{\xi}_{\mathcal{S}',t} \sim \tilde{\pi} = \sum_{s \in \mathcal{S}'} \boldsymbol{\lambda}_{s,t} \mathcal{N}(\boldsymbol{\mu}_{s,t}, \boldsymbol{\sigma}_{s,t})$
- Re-sample by using the SIR principle
- $\mathcal{G} = KClusterize(\tilde{\xi}_{\mathcal{S}',t})$
- $(\boldsymbol{I}_{s,t}, \boldsymbol{\Lambda}_{s,t}, \boldsymbol{M}_{s,t}, \boldsymbol{\Sigma}_{s,t}) = DataAssociation(\mathcal{G}, I_{s,t-1})$

end

As a difference with other clustering methods (e.g., *k-means*, *Hierarchical Clustering* or *QT-Clustering*), KClusterize does not require to know in advance the number of clusters, has a linear complexity, and all the obtained clusters reflect a Gaussian distribution.

Data association. An identity (i.e., a track number) has to be assigned to each object, by associating the new observations to the existing tracks. This is the crucial step for any tracking algorithm. The direction, the velocity and the position of the objects are the features involved in the association algorithm. We consider two moving tracked objects having the same direction if the angle between their trajectories is less than $10°$.

The data association step is further complicated by complete and partial occlusions, which can occur when objects are aligned with respect to the camera view or when they are close to each other. Our solution is to consider the collapsing tracks as a group, instead of tracking them separately. When two or more tracks have their bounding boxes moving closer to each other, the tracker saves their color histograms and starts considering them as a group - the histograms are used as models for re-identifying the objects when the occlusion phase is over. A group evolves taking into account both the estimated trajectory and the observations coming from the detector. When an occluded object becomes visible again, the stored histograms are used to re-assign the correct identification number, belonging to the corresponding previously registered track.

IV. Activity Forecasting From Noisy Visual Observations

Our objective in *activity forecasting* is the task of estimating future actions of moving agents from noisy visual

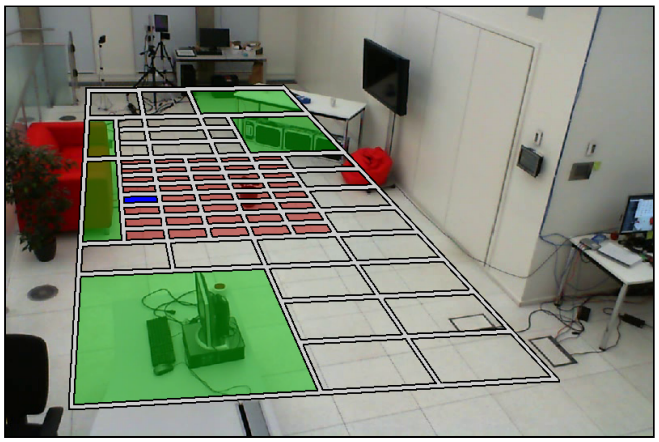


Fig. 3. Set of non-uniform grid representation of the environment. Dynamic regions (in red) are mapped with a set of dense grids, while the ones having few interactions (in green) are represented with sparse and small grids. The goal is highlighted by a blue cell of the grid.

input. We address this using an IRL procedure which works on the output of the PTracking approach. In a certain sense, the problem formulation ought to acknowledge that states are not known exactly, instead being estimated through a visual tracking process. So, the temporal modelling of activity may perhaps be described in the language of partially observable models. However, such models are rarely easy to work with, especially so when the goal is efficient real-time implementation on resource constrained robots. So, we proceed by making an assumption akin to 'certainty equivalence', modelling the temporal dynamics in terms of an MDP, which is fed the output of the PTracking algorithm which acts as state estimator. To the extent that the tracker maintains a fully probabilistic representation of objects and their motion and that the set of possible goals estimated by the IRL method can be incrementally grown, this is a useful compromise that could be further relaxed in future work.

A finite discrete-time MDP is a tuple $(\mathbf{S}, \mathbf{A}, \mathbf{P}_{sa}, \mathbf{R}, \gamma)$ where: $\mathbf{S}$ is a finite set of N states, $\mathbf{A} = \{a_1, a_2, \ldots, a_k\}$ is a set of k actions (i.e., North, West, South, East), $\mathbf{P}_{sa}(\cdot)$ are the state transition probabilities upon taking action a in a state s, $\mathbf{R} : \mathbf{S} \times \mathbf{A} \mapsto \mathbb{R}$ is the reinforcement function, bounded in absolute value by R_{max} and $\gamma \in [0, 1)$ is the discount factor. We adopt a standard IRL algorithm, due to Russel *et al.* [13], in this work. Their work makes use of the *policy optimality* theorem to derive a linear programming formulation of IRL as follows:

$$
\begin{aligned}
&\min \textstyle\sum_{i=1}^{N} -x_i + \lambda(r_i^+ - r_i^-) \\
&s.t. \\
&\begin{cases}
x_i \leq (\mathbf{P}_{a_*} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_*})^{-1}\mathbf{R} \\
\qquad\qquad\qquad\qquad \forall\, a \in \mathbf{A},\ i \in \{1, \ldots, N\} \\
x_i \geq 0 \qquad\qquad\qquad\qquad i \in \{1, \ldots, N\} \\
r_i = r_i^+ + r_i^- \qquad\qquad i \in \{1, \ldots, N\} \\
|\mathbf{R}_i| \leq R_{max} \qquad\qquad i \in \{1, \ldots, N\}
\end{cases}
\end{aligned}
\tag{3}
$$

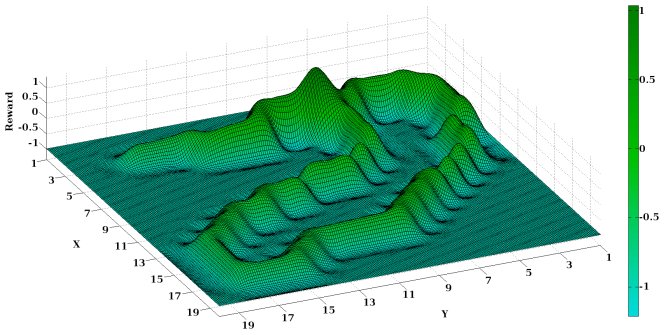


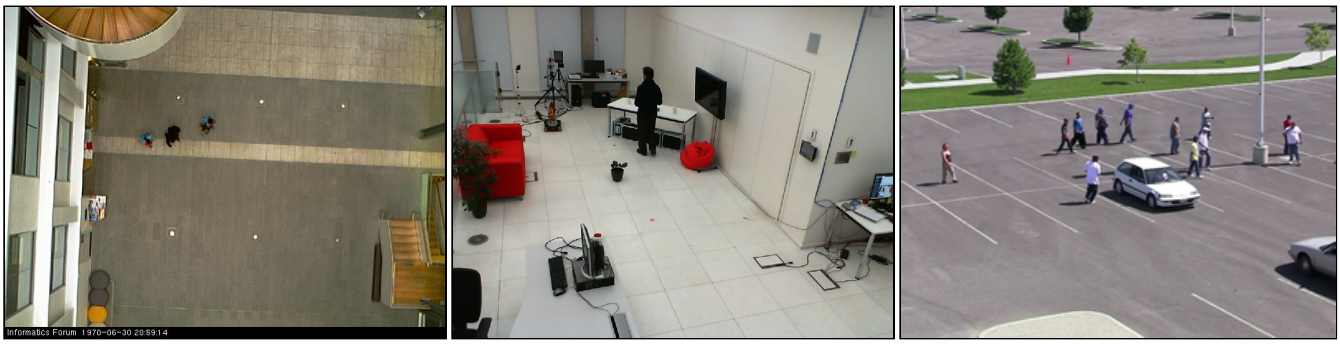
Fig. 4. Trajectory-based model learnt by solving the IRL problem defined in Eq. (3). The goal is represented by the function's global maximum.

Here, $\mathbf{P}_a$ denotes a NxN matrix in which element (i, j) gives the probability of transitioning to state j upon taking action a in state i, $\mathbf{P}_{a_*}$ denotes the current *optimal* policy, obtained by combining the optimal policy at the previous step and the output of the PTracking algorithm. In Eq. (3), the non-linear 1-norm operator $\|\mathbf{R}\|_1$ of the original problem stated by Russel *et al.* has been linearised by adding two more variables r_i^+ and r_i^- for every r_i variable with $i \in \{1, \ldots, N\}$. The non-linear operator $\min_{a \in \mathbf{A}}$, instead, has been linearised by introducing N new variables x_i where $i \in \{1, \ldots, N\}$. The linear programming problem defined in Eq. (3) can be easily and efficiently solved using standard techniques, such as the Simplex algorithm.

A. Inverse Reinforcement Learning Model

The result of solving the problem in Eq. (3) is the reward function (for each goal), that captures both interactions and movements of objects in the monitored environment. Future actions may be forecasted and suspicious trajectories (i.e., anomalies from the model obtained via IRL) recognised in advance. Since the problem defined by Eq. (3) is a discrete optimisation problem, we need to discretise continuous visual observations coming from the PTracking algorithm (see Section III). To this end, we propose a representation of the monitored environment based on a set of non-uniform grids, thus allowing for an effective and efficient representation of the monitored environment. In this manner, portions of the environment in which there are multiple interactions are represented by a set of dense grids, while parts of the environment that do not have particular interactions are described by sparse grids.

Grid update. First, a uniform grid mapping the monitored environment is constructed. Then, such a grid is periodically updated based on the information provided by the PTracking algorithm. High density locations are described richly by increasing the granularity of the grid mapping around that location, whereas parts of the environment having fewer interactions are described sparsely. By adopting the non-uniform grid representation, the proposed approach linearly scales with respect to the size of the environment as well as the computational resources required for solving the optimisation problem. An example of the non-uniform grid



(a) Informatics Forum (b) HRI Laboratory (c) VIRAT

Fig. 5. **Left:** very dynamic environment representing the main entrance to our Informatics Forum. **Center:** a simulation of a small home environment in our HRI laboratory with humans and robots. **Right:** VIRAT data set being also used by Ziebart *et al.* [1].

representation is depicted in Fig. 3, while a possible IRL model associated with it is shown in Fig. 4, in which the end goal is represented by the function's global maximum.

B. Activity Forecasting and Anomaly Detection

Our approach can be summarised in terms of the following key steps. The output of the optimisation process of Eq (3) is a set of G reward function models, one for each goal, $\mathcal{M}_s^G$ per agent s, where $s \in \mathcal{S}$. This allows us to generate a policy $\pi_s^{G_i}$ for each goal $G_i \in G$ per agent s. We can now build likelihoods over the set of goals G as follows.

Observed trajectory extraction. We gather object estimates from the PTracking algorithm considering an arbitrary temporal window ($5s$ for our experimental evaluation). Having acquired a set of trajectories $\mathcal{U}$, we ground each trajectory $u \in \mathcal{U}$ in every policy π_s^G.

Policy comparison. At this stage, we get the best fitting policy by comparing the target trajectory against a set of trajectories drawn from potential optimal policies. We do this using a combination of *Fréchet distance* and *cosine similarity*, to allow for the possibility that the target trajectory is merely a fragment of the overall optimal policy so that this notion of geometric similarity is one that better captures our notion of activity membership.

Goal prediction. We are finally able to predict in real-time the goal toward which each moving object is likely to be headed, by executing the policy that best matches the movement pattern of every object. It could happen that a trajectory $u \in \mathcal{U}$ does not match any model. This can happen for two main reasons. The first is that the trajectory fragment u is anomalous and refers to a suspicious activity pattern. The second, which we cannot always rule out, is that there may be a multitude of optimal policies that represent the activity class or that the environment has changed leading to new types of motion. The latter, however, can be recognised by analysing the foreground model provided by the detector algorithm because it will hugely differ from the background model learnt so far. All learnt models are discarded until new models are available.

Goal sampling. In the general case, goals may not be defined ahead of time. In a setting such as a home environment, goals could be associated with routine activity and, say, tools or objects frequently used by a human user. Here, static points of interest can be conjectured from a relatively inexpensive scene analysis, providing much needed contextual structure of the environment. However, in a dynamic scenario like an airport, train station or even a fast changing construction or rescue site, this process may be infeasible and the set of potential goals identified from surface level analysis may be very large. This problem is compounded by the lack of clear and persistently identifiable structure in these rapidly changing environments. In these cases, we could generate potential goals by analysing the information provided by a tracking system (i.e., analysing trajectories of all moving agents), and work with respect to this set.

V. Experimental Evaluation

The framework has been tested in three different environments: the main entrance to our Informatics Forum, our HRI laboratory and the VIRAT data set (Fig. 5). A quantitative comparison, depicted in Fig. 6, demonstrates how our proposed framework outperforms alternatives in terms of NLL - Eq. (4), including a state-of-the-art approach of Ziebart *et al.*, a Maximum Entropy Markov Model algorithm and a random walk baseline procedure.

Comparison metric. In each experiment, we have one demonstrated path, a sequence of states s_t and actions a_t, generated by all agents for a specific configuration of a scene. We compare the demonstrated path with the probabilistic distribution over paths generated by our IRL algorithm using the *Negative Log-Loss* (*NLL*) of a trajectory, as in [1], defined as follows:

$$NLL(s) = E_{\pi(a\,|\,s)}\Big[-\log \prod\nolimits_t \pi(a_t \,|\, s_t)\Big] \qquad (4)$$

The NLL represents the expectation of the log-likelihood of a trajectory s under a policy $\pi(a \,|\, s)$. This metric measures the probability of drawing the demonstrated trajectory from the learnt distribution over all possible trajectories.

Experimental setup. We compare our framework against the method proposed by Ziebart *et al.* in [1], a Maximum Entropy Markov Model based algorithm and a random walk baseline procedure in all scenarios. We use the same input data and the same state representation for the random walk baseline, the MEMM method and the proposed approach in

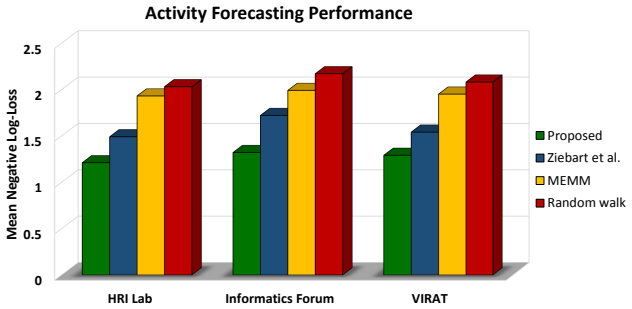


Fig. 6. Mean NLL of activity forecasting performance on chosen data sets. Lower values mean better activity forecasting accuracy.

all the chosen scenarios and for the Ziebart *et al.* approach on the HRI laboratory and Informatics Forum scenarios. We report, instead, the result taken from the corresponding paper of Ziebart *et al.* for the VIRAT data set.

HRI laboratory. This scenario simulates a small home environment (up to 5 moving agents) in which points of interest are extracted by analysing the tracking data. Robot and people's position and velocity estimates are provided by the distributed tracker using two overhead cameras, facing opposite directions but with overlapping fields of view over the environment. In high density scenarios, the activity forecasting task is made challenging by the limited collision-free space in proportion to the physical size of the agents involved. Such a constraint could force an agent to dramatically change bearing to avoid a dynamic obstacle while still approaching one's target goal.

Informatics forum. We evaluate our approach according to its performance in real-time tracking and activity forecasting in a natural human environment (up to 25 moving agents). This is challenging due to numerous aspects, such as the presence of agents with changing intentions, or agents that are navigating with other latent constraints (e.g., maintaining a spatial formation with respect to other agents). For this scenario, possible goals have been conjectured by analysing the information provided by the tracking algorithm. Our results show that, our activity forecasting algorithm provides accurate beliefs over the possible set of goals.

VIRAT. The data set is designed to be realistic, natural and challenging for video surveillance domains in terms of its resolution, background clutter, diversity in scenes and human activity/event categories. In order to fairly compare our IRL approach against the state-of-the-art method of Ziebart *et al.*, we choose goals as done in [1].

Discussion. In this work, we have access to trajectories of normal behaviours that are used for the initial generation of an MDP model through IRL, for each agent. Such a model describes the *preferred* paths of an agent moving toward a certain goal. The generated model is independent in terms of agent's velocity, hence a prediction of future agent motions, having an arbitrary velocity with respect to the observed one, is still possible by applying the IRL model. In the case of dramatic changes in the environment dynamics, the IRL model becomes essentially unusable due to this considerable variation in the structure of the environment. Therefore, an updated IRL model, taking into account these new changes, is needed before forecasting agent intentions. This suggests that the foregoing description is to be viewed as a template of a framework that can be further enhanced with lifelong and continual learning towards efficient activity forecasting for a practical social robot.

VI. Conclusions

We presented a novel framework for estimating the future movement intentions of goal-oriented agents in an interactive multi-agent setting. We achieve this by combining Inverse Reinforcement Learning with a Markov Decision Process model of motion, and a distributed multi-camera tracking algorithm. The resulting reward functions represent the agent's set of possible future activities, on which forecasts are made through a probability distribution function via comparison with real-time observed agent behaviour. This method is evaluated for accuracy and robustness in dense and dynamic environments with autonomously planning robots and pedestrians. Our results show that this is an effective and computationally efficient alternative to models that depend either on offline training of pedestrian trajectory models or on physical scene features and prior knowledge of goals.

References

[1] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision*, 2012, pp. 201–214.

[2] A. Bordallo, F. Previtali, N. Nardelli, and S. Ramamoorthy, "Counterfactual reasoning about intent for interactive navigation in dynamic environments," in *International Conference on Intelligent Robots and Systems*, 2015, pp. 2943–2950.

[3] F. Previtali and L. Iocchi, "PTracking: distributed multi-agent multi-object tracking through multi-clustered particle filtering," in *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2015, pp. 110–115.

[4] F. Previtali, G. Gemignani, L. Iocchi, and D. Nardi, "Disambiguating localization symmetry through a multi-clustered particle filtering," in *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2015, pp. 283–288.

[5] D. D. Bloisi, F. Previtali, A. Pennisi, D. Nardi, and M. Fiorini, "Enhancing automatic maritime surveillance systems with visual information," *Transactions on Intelligent Transportation Systems*, pp. 1–10, 2016.

[6] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: a review," *ACM Computing Surveys*, vol. 43, no. 3, p. 16, 2011.

[7] J. C. Nascimento, M. A. T. Figueiredo, and J. S. Marques, "Trajectory classification using switched dynamical hidden Markov models," *Transactions on Image Processing*, vol. 19, no. 5, 2010.

[8] A. Jaimes and N. Sebe, "Multimodal human-computer interaction: a survey," *Computer Vision and Image Understanding*, vol. 108, no. 1, pp. 116–134, 2007.

[9] A. Ekin, A. M. Tekalp, and R. Mehrotra, "Automatic soccer video analysis and summarization," *Transactions on Image Processing*, vol. 12, no. 7, pp. 796–807, 2003.

[10] C. G. M. Snoek and M. Worring, "Concept-based video retrieval," *Foundations and Trends in Information Retrieval*, vol. 2, no. 4, 2008.

[11] D. L. Vail, M. M. Veloso, and J. D. Lafferty, "Conditional random fields for activity recognition," in *International Conference on Autonomous Agents and Multiagent Systems*, 2007, p. 235.

[12] L. Wang, Y. Qiao, and X. Tang, "Latent hierarchical model of temporal structure for complex activity classification," *Transactions on Image Processing*, vol. 23, no. 2, pp. 810–822, 2014.

[13] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning*, 2000, pp. 663–670.